BOJ 16432 풀이

떡장수와 호랑이

Approach

Approach

3			
3	1	2	3
2	1	2	
2	2	3	

1	1	1						
1	1							
	1	1						

1. 떡을 index로 표시한다 (map[][])

➔ 각 경우의 수를 돌아보면서 정답이 있다면 출력한다.
: DFS를 이용

Approach

2. 조건체크: 어제 먹인 떡은 안됨

1. check() : DFS가 가장 깊이 도착하면 체크

불필요하게 DFS돌려야 됨 + for문도 돌려야 됨 등등
암튼 별로임

2. Pre(전날 준 떡)를 global variable로 저장

DFS돌때마다 바뀌어서 값이 제대로 안 나옴(78%에서 틀림)

3. Pre(전날 준 떡)를 parameter로 전달

굿

CODE

Variable

```
N=Integer.parseInt(br.readLine());
map=new int[N+1][10];
result=new int[N+1];
visited=new boolean[N+2][10];
```

- N: day
- Map: day별 떡 표시
- Result: [1]에는 1일에 준 떡, [2]에는 2일에 준 떡…
- Visited: 방문처리

Main()

```
for(int i=1;i<N+1;i++) {
    st=new StringTokenizer(br.readLine());
    int riceCake=Integer.parseInt(st.nextToken());
    for(int j=0;j<riceCake;j++) {
        map[i][Integer.parseInt(st.nextToken())]=1;
    }
}

DFS(1, 0);

bw.append("-1");
bw.flush();bw.close();br.close();
```

떡의 종류를 index로 하여 map에 1로 표시해 둠.

첫날부터 DFS 시작 → DFS(1, pre)

DFS()

```java
public static void DFS(int day, int pre) throws Exception{
    if(day==N+1) {
        for(int i=1;i<N+1;i++) {
            bw.append(result[i]+"\n");
        }
        bw.flush();
        System.exit(0);
    }

    for(int i=1;i<10;i++) {
        if(map[day][i]==1 && !visited[day+1][i] && i!=pre) {
            visited[day+1][i]=true;
            result[day]=i;
            DFS(day+1, i);
        }
    }
}
```

DFS돌다가 depth끝까지 옴
→ result 출력

1. 어제 떡 / 오늘 떡 다름
2. 해당 떡이 있음
3. 다음 DFS 진행할 방향에 방문X
→ DFS 진행 + result에 저장